Expert Systems with Applications 38 (2011) 249–259

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

# A software model to prototype ant colony optimization algorithms

Roberto Fernandes Tavares Neto [*], Moacir Godinho Filho

*Industrial Engineering Department, Federal University of São Carlos (UFSCar), Rodovia Washington Luis, Km 235, São Carlos – SP, Brazil*

ARTICLE INFO

*Keywords:*
Ant colony system
Software model
ACS
AS
MMAS

ABSTRACT

The study of multi-agent systems usually begins by implementing a base-algorithm, which is changed as required by the aim of the research. In this context, carrying out different algorithms, which have already been established, is not a trivial task as it requires implementing these algorithms. This paper presents a software model that allows one to prototype variations of the Ant Colony Optimization metaheuristic. This model can be used to avoid implementations in duplicity, allowing, with less effort, the generation of different algorithms to be used on the same problem. Results shown that, specially for more elaborated algorithms, the adoption of the proposed software model reduce significantly the coding effort required.

Crown Copyright © 2010 Published by Elsevier Ltd. All rights reserved.

## 1. Introduction

The Ant System (AS) algorithm was introduced in 1991 by Colorni et al. Since then, a lot of research has been done by applying ants' behavior to classic and new problems reported in the literature. Gambardella and Dorigo (1996) introduced the Ant Colony System (ACS) algorithm improving the original AS algorithm and producing better results to solve combinatory problems.

Several researchers have adapted the heuristics proposed by Gambardella and Dorigo (1996) to specific problems. ACS was first used to solve the classical Traveling Salesman Problem (TSP), symmetric and asymmetric, (Dorigo, Maniezzo, & Colorni, 1996; Gambardella & Dorigo, 1996). After that, several other applications were developed, such as the Sequential Ordering Problem – SOP, (Gambardella & Dorigo, 2000), the Capacitated Vehicle Routing Problem – CVRP, originally solved by Bullnheimer, Hartl, and Strauss (1997), scheduling problems (Bauer, Bullnheimer, Harlt, & Strauss, 2000; Blum, 2002; Stützle & Hoos, 2000) among others. For more information, Dorigo and Stutzle (2004) present a sample of more than 60 applications of AS and ACS.

According to Stutzle and Linke (2000), all of this research is based on the same core algorithm with small modifications incorporated in order to include the new characteristics, maintaining the basic functions of the ant colony algorithm. However, for each implementation, a different computational tool is used, so it is common to find mentions of codes written in C, C++, JAVA or even using mathematical processing tools such as Matlab, depending on the availability of resources and technical preferences of each researcher. This lack of standardization in code structure is a problem usually found for researchers who want to apply and to compare different algorithms.

Within this context, this paper proposes and implements a software model that facilitates information exchange between researchers who use the AS algorithm and its variations. Analyzing the necessary changes in the original core of the AS for the implementation of the different heuristics, it is possible to establish a relationship between them and their classification. Using the proposed software model, the creation of new algorithms from the initial AS core is made easier, since all the support functions and the basic behavior is already validated. At the same time, implementing different algorithms with the same input data representation model facilitates the evaluation of the use of techniques that are already available in different classes of problems.

Similar studies can be found in the literature, for example Andreatta, Carvalho, and Ribeiro (2002), who present a software model for the creation of local search heuristics; Laguna (1997), who present a software for optimization with genetic algorithms; and the projects "Genetic Algorithms Framework" (GA-FORK, 2008) and "COIN" (Hunsaker, 2008) which provide respectively a multiplatform software model to solve problems using genetic algorithms and a computational infrastructure for operational research problems. Nonetheless, no software model applied to design prototypes of systems based on the ant colony metaheuristic was found in the literature.

In order to achieve the proposed goals, the algorithms Ant-Cycle, Ant-Density, and Ant-Quantity, proposed by Colorni, Dorigo, and Maniezzo (1991) and Dorigo et al. (1996), were used. The results obtained from surveys (Dorigo & Blum, 2005; Stovba, 2005) will be the basis to show how the system proposed can be used for the implementation of variations of the single colony AS. In addition, using the paper of Ellabib, Calamai, and Basir (2007) we demonstrate how to implement AS multiple colony algorithms using the software model presented. Results show that when the

[*] Corresponding author. Tel.: +55 16 3351 9540; fax: +55 16 3351 8240.
*E-mail addresses:* tavares@dep.ufscar.br (R.F. Tavares Neto), moacir@dep.ufscar.br (M. Godinho Filho).

0957-4174/$ - see front matter Crown Copyright © 2010 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.06.054

proposed model is used, the implementation effort could be reduced significantly. As an example, the implementation of the Max–Min Ant System (Stützle & Hoos, 2000) required only three simple methods to be written while without the software model, 40 methods were necessary to be written.

This article is structured as follows; Section 2 presents the definition of the AS algorithm, and its major characteristics and variations. Section 3 presents the software model proposed. Section 4 presents the project choices for the implementation of the structure introduced, the necessary changes for the implementation of some algorithms described in Section 2, and also the results of a set of tests to validate the implemented procedures. The last section presents some final considerations and ideas for further research.

## 2. The ant colony optimization algorithm

The ant colony optimization algorithm is based on the concept of multi-agent systems. As agent, this paper will consider the definition given by Frankling and Graesser (1996): *"a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future*". In ant colony algorithms, the agents' environment is represented by a graph consisting of different numerical information: a fixed one, established with the definition of the problem, and a variable one, which is modified within the algorithm execution. They do not depend on each other and are related only to the arcs connecting the nodes i and j in the graph. Some examples of fixed information are the distance between cities (for modeling problems such as the TSP) and time required for operations (for scheduling problems), among others. The modified information along the algorithm is related to "artificial pheromones" values making an analogy with the pheromones emitted by real ants while moving.

According to Dorigo et al. (1996), an artificial ant can be defined as a computational agent with the following characteristics:

1. It exists in an environment represented mathematically as a graph: an ant always occupies a node in a graph which represents a search space. This node is called *nf*.
2. It has an initial state.
3. Although it cannot sense the whole graph, it can collect two kinds of information about the neighborhood: firstly, the weight of each trail linked to *nf*; and secondly, the characteristics of each pheromone deposited on this trail by other ants of the same colony.
4. Moves toward a trail $c_{ij}$ that connects nodes $i$ and $j$ of the graph.
5. Also, it can alter the pheromones of the trail $c_{ij}$, in an operations often called on literature as "deposit of pheromone levels".
6. It can sense the pheromone levels of all $c_{ij}$ trails that connects a node $i$.
7. It can determine a set of "prohibited" trails.
8. It presents a pseudo-random behavior enabling the choice among the various possible trails.
9. This choice can be (and usually is) influenced by the level of pheromone.
10. It can move from node $i$ to node $j$.

Additionally, if the ant has more information about the problem, such as the number of elements expected in the final solution, the computational implementation can be simplified. In this case, there is a computational gain allocating the memory of the vector representing a fixed size response instead of recalculating its size in every step of building an individual response. The knowledge of the problems characteristics also enables the implementation of a set more sophisticated rules used to determine the next movement (for example, see Bauer et al., 2000).

The behavior of the AS, as shown in Chart 1, is based on the concept of emerging intelligence: the problem is not solved directly by one or two agents, but, instead, it is a result of the behavior of the various interactions between the agents and their environment. This can occur by concentrating efforts to solve the problem within the search space, where it is supposed to have high quality solutions. According to Dorigo, Bonabeau, and Theraulaz (2000), real ants use pheromones for communication. This mechanism is the following: while a single ant moves randomly, an ant which finds a pheromone trail has a higher probability to follow this trail. In addition, when this ant chooses this trail, it deposits more pheromone into the trail which increases the probability of another ant follow the same trail. According to Blum and Dorigo (2004a), algorithms based on ant system try to imitate this behavior creating artificial pheromones which are continually updated to increase the probability of an ant to choose the best trail.

In Chart 1, there are three fundamental elements for algorithms based on ant colonies (shown in steps 5, 6, and 8 respectively):

1. Probability for an ant to choose a trail (shown in the literature as "transition rule", shown in step 5).
2. Pheromone local trail update when building the solutions (step 6).
3. Pheromone global update rule, which has an evaporation rule (negative reinforcement) and a pheromone deposit rule to be applied after building an individual solution (positive reinforcement, step 8).

As stated before, pheromones can be updated twice: firstly during the movement of the ants (before finalizing the solution building), and later, after all ants have built their solutions.

It is worth mentioning that the behavior predicted by steps 6 and 8 in Chart 1 is crucial for the whole agent communication process since there is no direct communication, but a synergetic communication used for coordinating the ants' actions. This indirect communication occurs by incrementing and decrementing the pheromone levels, as shown in Fig. 1. This figure shows two routes that connects two nodes ("origin and destination"). At first, the pheromone level of both routes (represented by the length of the corresponding arrows) is equalized (Fig. 1a). In this case, there is an equal chance of choice between the two routes. Therefore, the same number of ants will move through them. Nevertheless, it takes longer for ants that choose the longest route to return, causing more evaporation of the pheromone they deposited. Generally, the pheromone deposit rules are designed to penalize the positive reinforcement in longer routes. In both cases, the pheromone level of the longer routes decreases in relation to the pheromone level of the shorter trail (Fig. 1b). After some iterations, the shorter route

```
1. Initialize
2. Execute /* at this level, each loop is called iteraction*/
3.         Each ant is positioned on the initial node
4.         Execute /*at this level, each loop is called step*/
5.              Each ant applies a transition state rule to generate the solution
6.              Apply a local update rule
7.         Until all ants have build a complete solution
8.         Apply a global pheromone update rule
9. Until a stop criterion is satisfied
```

**Chart 1.** Basic structure of the AS algorithm (source: Dorigo and Gambardella, 1997).